art - Idea #18942

Possible way around SIP and allowing use of DYLD_LIBRARY_PATH on macOS?

02/09/2018 09:49 AM - Ben Morgan

Status:	Feedback	**Start date:**	02/09/2018
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Experiment:	-	**SSI Package:**	

Description

This has probably been looked at already, but I thought I'd open an Idea ticket couldn't find an existing ticket/discussion on it.

From the CVMFS distribution of larsoft/art (so art v2_09_06), I've been able to run art and at least the hello.fcl example from toyExperiment on a High Sierra box **with SIP enabled** as follows

- Use homebrew to install openssl *and* bash
- Start the Homebrew supplied bash
- Source the setups script in this shell session, setup art and toyExperiment, and run - this can find all the libraries and exit successfully.

There are a lot of caveats here:

1. It's a hack, as I'm using art's Darwin 16 binaries on 17 (via UPS_OVERRIDE), but I'm guessing there's minimal use of system libs that might cause incompatibilities...
2. ... so it's purely checking the SIP/DYLD_LIBRARY_PATH issue, no other macOS issues.
3. It'll only work if SIP has the default configuration (it relies on the shell, and programs started from it, being outside the SIP protected areas)
4. It can fail if programs are run inside a script with a shbang that is !#/bin/bash (e.g. cet_test_exec), but this can be fixed by using #!/usr/bin/env <program>.

I was also able to checkout, setup_for_development and buildtool at least cetlib_except apart from the tests (per that last point), but the tests can be run directly and pass.

It's not a long term solution, and may have been tried and dismissed before, but might offer a temporary workaround without needing to turn off SIP, at least for simple running of art/larsoft (likely many development/debug issues that I haven't encountered yet). I think the use of rpath that Spack/SpackDev may offer, plus dedicated lookup paths for Art plugins etc like the solution in #17900, should be more robust long term.

History

#1 - 02/12/2018 12:18 PM - Kyle Knoepfel

This is an encouraging step forward--we had not considered this approach yet. We would be thrilled for you to continue pursuing this route with the other packages that are part of the *art* suite. We are happy to review and accept any patches you have that result in a completely functioning system. Thanks!

#2 - 02/12/2018 12:19 PM - Kyle Knoepfel

- Status changed from New to Feedback

#3 - 02/20/2018 10:27 AM - Ben Morgan

Kyle Knoepfel wrote:

> This is an encouraging step forward--we had not considered this approach yet. We would be thrilled for you to continue pursuing this route with the other packages that are part of the *art* suite. We are happy to review and accept any patches you have that result in a completely functioning system. Thanks!

Will do, as far as I can. Based on the results of few further tests, I think the fundamental patch will be to use the noted #!/usr/bin/env <program> form of shebang so I can go ahead and start working through the packages to add these if that's acceptable?

I've also identified one issue with CTest in that it seems to run tests using a system shell/process, so even if the cet_exec_test wrapper script starts a non-SIP bash, the environment has already been stripped. This can be worked around in cetbuildtools/CetTest by using either an internal env var (i.e.

not requiring any UPS setup) or additional command line argument to transmit the path to cet_test_exec, which then sets DYLD_LIBRARY_PATH internally. The command line arg would require the fewest changes to downstream clients, but let me know if there's a preferred method here.

With those fixes, development on Mac seems to work o.k. and I've been able to get everything up to and including art (v2_10_01) building and running with all tests passing natively on High Sierra with latest AppleClang/Xcode, at least for C++14.

#4 - 02/26/2018 11:49 AM - Kyle Knoepfel

> Will do, as far as I can. Based on the results of few further tests, I think the fundamental patch will be to use the noted #!/usr/bin/env <program> form of shebang so I can go ahead and start working through the packages to add these if that's acceptable?

Our only concern with /usr/bin/env is that we have some cases where we use

```
#!/usr/bin/perl -w
```

for which we are not aware of a replacement when using /usr/bin/env.

> I've also identified one issue with CTest in that it seems to run tests using a system shell/process, so even if the cet_exec_test wrapper script starts a non-SIP bash, the environment has already been stripped. This can be worked around in cetbuildtools/CetTest by using either an internal env var (i.e. not requiring any UPS setup) or additional command line argument to transmit the path to cet_test_exec, which then sets DYLD_LIBRARY_PATH internally. The command line arg would require the fewest changes to downstream clients, but let me know if there's a preferred method here.

Could you spell out in greater detail your first option? We think the second option is plausible, but we'd like to understand better the first.

> With those fixes, development on Mac seems to work o.k. and I've been able to get everything up to and including art (v2_10_01) building and running with all tests passing natively on High Sierra with latest AppleClang/Xcode, at least for C++14.

This is good news.

#5 - 03/02/2018 11:15 AM - Ben Morgan

Kyle Knoepfel wrote:

> > Will do, as far as I can. Based on the results of few further tests, I think the fundamental patch will be to use the noted #!/usr/bin/env <program> form of shebang so I can go ahead and start working through the packages to add these if that's acceptable?
>
> Our only concern with /usr/bin/env is that we have some cases where we use
>
> [...]
>
> for which we are not aware of a replacement when using /usr/bin/env.

At least for Perl, my understanding is that use warnings; covers this (but I'm not a Perl guru...), and similarly for other languages. Use of env isn't quite as straightforward as I thought though, the issue being that it is a SIP program itself, hence the issues I was having with CTest.

> > I've also identified one issue with CTest in that it seems to run tests using a system shell/process, so even if the cet_exec_test wrapper script starts a non-SIP bash, the environment has already been stripped. This can be worked around in cetbuildtools/CetTest by using either an internal env var (i.e. not requiring any UPS setup) or additional command line argument to transmit the path to cet_test_exec, which then sets DYLD_LIBRARY_PATH internally. The command line arg would require the fewest changes to downstream clients, but let me know if there's a preferred method here.
>
> Could you spell out in greater detail your first option? We think the second option is plausible, but we'd like to understand better the first.

I've prototyped the env var based option as follows (NB: links are to the prototype code on GitHub for clarity):

1. In CetTest, cache any existing value for the dynamic loader path on the first, and only the first, CMake run: https://github.com/drbenmorgan/cetbuildtools2/blob/f1bec68040b16c59d1d382e22164afedbd48a177/Modules/CetTest.cmake#L264-L280
2. Use CMake's ENV command to set "CETD_LIBRARY_PATH" in the CMake runtime environment as a universal dynamic loader path: https://github.com/drbenmorgan/cetbuildtools2/blob/f1bec68040b16c59d1d382e22164afedbd48a177/Modules/CetTest.cmake#L281-L285
3. Default the test environment to hold this variable and value: https://github.com/drbenmorgan/cetbuildtools2/blob/f1bec68040b16c59d1d382e22164afedbd48a177/Modules/CetTest.cmake#L287-L293, and similarly when CLEAR ing the environment: https://github.com/drbenmorgan/cetbuildtools2/blob/f1bec68040b16c59d1d382e22164afedbd48a177/Modules/CetTest.cmake#L298-L311
4. In cet_test_exec, set the dynamic loader path appropriate for the system using CETD_LIBRARY_PATH and the system loader path, if these exist. Note that the script is still using #!/bin/bash, so possible the @env isn't needed...

Caveats are:

1. CMake must be non-SIP, and run from an environment in which any required DYLD_LIBRARY_PATH (e.g. from setup_for_development) isn't stripped.
2. The dynamic loader path, as seen by the build/tests, can't be updated after initial CMake (but I don't know if there's a use case for this).
3. If the test program run by cet_test_exec is a shell script itself, that script may also need the "shim" section to forward the environment if it runs programs that rely on the dynamic loader. So far, this only seems to be a pattern used in the art product.

The first two points can be mitigated if the dynamic loader path is only needed by the project's tests as it can be set directly using cet_test_env and genexes. That's also been prototyped up to the art level, e.g.
https://github.com/drbenmorgan/fnal-cetlib/blob/48bd81e25bbb4fa5ebf847fd509dcd8c483dbb30/cetlib/test/CMakeLists.txt#L11-L19 .

That's the envvar method - using a command line arg to cet_test_env would be the same internally in CetTest, just that "CETD_LIBRARY_PATH" would be passed via an argument.

> With those fixes, development on Mac seems to work o.k. and I've been able to get everything up to and including art (v2_10_01) building and running with all tests passing natively on High Sierra with latest AppleClang/Xcode, at least for C++14.

This is good news.

So far the prototype linked above hasn't broken anything, but I still have to go through the art test suite again. Assuming that is o.k., is toy-experiment still the main art demo? I'm confident that the prototype will build/test/install art o.k. but the next step would be to try it from the user side for additional gotchas...

#6 - 03/05/2018 04:04 PM - Ben Morgan

Ben Morgan wrote:

> 1. In cet_test_exec, set the dynamic loader path appropriate for the system using CETD_LIBRARY_PATH and the system loader path, if these exist. Note that the script is still using #!/bin/bash, so possible the @env isn't needed...
>
> Caveats are:
>
> 1. CMake must be non-SIP, and run from an environment in which any required DYLD_LIBRARY_PATH (e.g. from setup_for_development) isn't stripped.
> 2. The dynamic loader path, as seen by the build/tests, can't be updated after initial CMake (but I don't know if there's a use case for this).
> 3. If the test program run by cet_test_exec is a shell script itself, that script may also need the "shim" section to forward the environment if it runs programs that rely on the dynamic loader. So far, this only seems to be a pattern used in the art product.
>
> The first two points can be mitigated if the dynamic loader path is only needed by the project's tests as it can be set directly using cet_test_env and genexes. That's also been prototyped up to the art level, e.g.
> https://github.com/drbenmorgan/fnal-cetlib/blob/48bd81e25bbb4fa5ebf847fd509dcd8c483dbb30/cetlib/test/CMakeLists.txt#L11-L19 .

One way, using the env var method, to solve the third point about the test executable being a shell script that requires DYLD_LIBRARY_PATH (e.g. some of the art tests are shell scripts that run art) is as follows:

1. Rejig the environment forwarding in cet_test_exec so that it also appends to PATH the location of cet_test_functions.sh:
https://github.com/drbenmorgan/cetbuildtools2/blob/270079fc0097cad8b24444582c1db608bf069caf/Modules/cet_exec_test#L613-L644
2. In cet_test_functions.sh, repeat the "shim" section for the dynamic loader path:
https://github.com/drbenmorgan/cetbuildtools2/blob/270079fc0097cad8b24444582c1db608bf069caf/Modules/cet_test_functions.sh#L50-L69
3. In any shell script tests, source this script before anything else. In a few cases this is done already and in others it doesn't seem to have any adverse effects, at least for all the shell script tests in the art product.

At least for build, install, and use of installed libs from CMake/make of cetlib_except through to art (art_suite 2.10.1 tag(s)), this fixes all remaining issues with compiling and running tests in a SIP enabled environment.

#7 - 03/12/2018 11:39 AM - Kyle Knoepfel

Thanks, Ben, for all your work on this.

In full honesty, we have some concerns that the full system may run into a complication that has not yet been exposed. To ease our concern, we would like to see a full MRB build of an experiment that uses *art*. We understand that may be a tall order, so starting with toyExperiment would be a great first step.

Unfortunately, toyExperiment does not have any unit tests, but we are hoping to provide some soon.

#8 - 03/12/2018 01:07 PM - Ben Morgan

Kyle Knoepfel wrote:

> Thanks, Ben, for all your work on this.

> In full honesty, we have some concerns that the full system may run into a complication that has not yet been exposed. To ease our concern, we would like to see a full MRB build of an experiment that uses *art*. We understand that may be a tall order, so starting with toyExperiment would be a great first step.

MRB is certainly an area I haven't gone near yet, as it's not a tool I really use. Naively I'd expect the same technique to work, but if you could enumerate the concerns in more detail as to where you foresee issues arising, I can try some things out at least in art_suite and see what happens. However, I think multirepo working might be better targeted at spack/dev due to the full decoupling between config management and build system that promises.

Modulo the issue reported in #19305, toyExperiment builds and the basic helloworld scripts are runnable on with manual setting of DYLD_LIBRARY_PATH to the local build area, though more testing is needed.

> Unfortunately, toyExperiment does not have any unit tests, but we are hoping to provide some soon.

Thanks!

#9 - 03/14/2018 11:48 AM - Ben Morgan

For information, a full build and install of art that works natively on macOS with SIP enabled system and not requiring UPS is "test flight ready". It's packaged through homebrew/linuxbrew+Docker for now as that was the simplest/fastest way to demo it - I'm fully aware that this does **not** cover the mrb/full environment issues noted above, nor usage/install/configuration of multiple versions/compilers/standards, nor use by experiments etc, etc.

It's simply presented to demo that art is usable with SIP enabled and/or outside of a UPS managed environment - so *might* provide some input to the spack work (given it packages in home/linuxbrew, spack packages *should* be relatively straightforward.

Details and install/use instructions for Mac and Linux/Docker here: https://github.com/drbenmorgan/homebrew-art_suite

#10 - 05/21/2018 05:41 AM - Ben Morgan

A quick query - with CetModules now being developed actively and the initial structure in cetlib_except, would "Pull Requests" to CetModules implementing the functionality described above be appropriate? It would seem better to attempt/try the fixes here upfront, rather than try and backport to cetbuildtools/mrb, depending on your plans/timescales of course.